NDDS®

Network Data Delivery Service

The Real-Time Publish-Subscribe Connectivity Solution

Getting Started Guide

NDDS Version 3.0m

Copyright © 1996-2003 Real-Time Innovations, Inc.
All rights reserved.
Printed in U.S.A. First printing.
August 2003.

Trademarks

Real-Time Innovations, Constellation, ControlShell, NDDS, RTI, StethoScope, WaveScope, WaveSurf, and WaveWorks are either trademarks or registered trademarks of Real-Time Innovations, Inc.
ARM7TDMI and SA-110 are trademarks or registered trademarks of ARM Limited.
Intel and Pentium are registered trademarks of Intel Corporation.
PowerPC is a registered trademark of International Business Machines Corporation.
Linux is a registered trademark of Linus Torvalds.
LynxOS is a registered trademark of LynuxWorks, Inc.
Microsoft, Windows, Windows NT, and Visual C++ are registered trademarks of Microsoft Corporation in the United States and/or other countries.
MIPS is a registered trademark of MIPS Technologies Inc.
Motorola is a registered trademark of Motorola, Inc.
UNIX is a registered trademark of The Open Group in the U.S. and other countries.
SPARC, SPARCV9, and UltraSPARC are registered trademarks of SPARC International, Inc.
Sun and Solaris are either trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.
Diab, pSOS, Tornado, VxWorks, and Wind River are either trademarks or registered trademarks of Wind River Systems, Inc.
All other trademarks used in this document are the property of their respective owners.

Copy and Use Restrictions

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form (including electronic, mechanical, photocopy, and facsimile) without the prior written permission of Real-Time Innovations, Inc. The software described in this document is furnished under and subject to the RTI software license agreement. The software may be used or copied only under the terms of the license agreement.

Technical Support

Real-Time Innovations, Inc.
155A Moffett Park Drive
Sunnyvale, CA 94089
Phone: 408-734-4200
Fax: 408-734-5009
E-mail: support@rti.com
Web Site: http://www.rti.com

Contents

Tables ..... **vii**

Welcome to NDDS ..... **ix**

Available NDDS Documentation ix
Reading Guide x
Conventions xi
Typographical Conventions xi
Pathnames xi
Environment Variables xi
Names of Supported Platforms xi
Coding Conventions xii

1 Release Notes for NDDS 3.0m **1-1**

1.1 System Requirements 1-1
1.1.1 Supported Operating Systems and Compilers 1-1
1.1.2 Disk and Memory Usage 1-4
1.1.3 Networking Support 1-4
1.2 Compatibility with Other RTI Products 1-4
1.3 What is New or Changed in NDDS 3.0m since NDDS 3.0i 1-5
1.3.1 New in NDDS 3.0m 1-5
1.3.2 Removed or Deprecated in NDDS 3.0m 1-5
1.3.3 Changes in NDDS 3.0m 1-5
1.4 Known Issues in NDDS 3.0m 1-6
1.4.1 Feature Limitations on Specific Operating Systems 1-6

1.4.2 Minimum Separation Problem with Multiple Multicast Subscriptions ... 1-6
1.4.3 Unsupported Data Types in nddsgen ... 1-7
1.4.4 nddsManager Holds onto Sockets Opened by the Spawning Application ... 1-7
1.4.5 Polled Subscription Deadline Behavior ... 1-7
1.5 Removed as of NDDS 3.0i ... 1-8
1.5.1 NDDS_UPDATE_OF_OLD_DATA Event ... 1-8
1.6 Removed as of NDDS 3.0g ... 1-8
1.6.1 Use of "u_*" types in .x Files Not Recommended ... 1-8
1.6.2 Deprecated C Creation Functions ... 1-8

2 Installation and Setup ... 2-1

2.1 Overview ... 2-1
2.2 Downloading NDDS ... 2-2
2.3 Installing on UNIX Systems ... 2-3
2.3.1 Testing Your Installation ... 2-4
2.4 Installing on Windows Systems ... 2-5
2.4.1 Installing Additional Target Libraries on Windows Systems ... 2-7
2.4.2 Testing Your Installation ... 2-7
2.5 Setup ... 2-8
2.5.1 Setting NDDS_PEER_HOSTS on UNIX Systems ... 2-9
2.5.2 Setting NDDS_PEER_HOSTS on Windows Systems ... 2-9
2.5.3 Setting NDDS_PEER_HOSTS on VxWorks Systems ... 2-10

3 Compiling NDDS Applications ... 3-1

3.1 Compiling UNIX Applications ... 3-1
3.1.1 Compiling ... 3-2
3.2 Compiling Windows Applications ... 3-5
3.2.1 Setting the NDDSHOME Environment Variable ... 3-5
3.2.2 Compiling ... 3-7
3.3 Compiling VxWorks Applications ... 3-8

Index ... Index-1

Tables

Table 1	Typographical Conventions	xi
Table 1.1	Supported Target Architectures	1-2
Table 2.1	NDDS Components Window	2-6
Table 3.1	UNIX Architecture-Specific Build Settings	3-3
Table 3.2	UNIX Architecture NDDS Build Settings for Shared Libraries	3-4
Table 3.3	UNIX Architecture NDDS Build Settings for Static Libraries	3-4
Table 3.4	Windows Architecture NDDS Build Settings for Static Libraries	3-8
Table 3.5	Windows Architecture NDDS Build Settings for Dynamic Libraries	3-8
Table 3.6	Architecture-Dependent Compiler Settings for VxWorks 5.4/Tornado 2.0.x	3-9
Table 3.7	Architecture-Dependent Compiler Settings for VxWorks 5.4/Tornado 2.1.x	3-9
Table 3.8	Architecture-Dependent Compiler Settings for VxWorks 5.5/Tornado 2.2	3-10

Welcome to NDDS

Welcome to *Network Data Delivery Service* (*NDDS*®)—the real-time network middleware from Real-Time Innovations, Inc. *NDDS* provides an easy way to build applications that communicate over a network. With *NDDS*, applications can transfer information between different architectures and operating systems without any changes to the application code. *NDDS* is designed for real-time applications that need control over timing and memory usage, have low latency requirements, or need high robustness to network changes or failures.

Available NDDS Documentation

NDDS documentation includes the following documents in PDF format, which is convenient for printing:

- ❏ This *Getting Started Guide* (**GettingStarted.pdf)**.
- ❏ The *NDDS User's Manual* (**Manual.pdf**).
- ❏ The *NDDS Tutorial (***NDDSTutorial.pdf)**.

This *Getting Started Guide* can be downloaded separately from the *NDDS* distribution. When you install *NDDS*, all three PDF documents are automatically placed in the **pdf/** directory of your *NDDS* installation.

NDDS documentation also includes the following in HTML format:

- ❏ The *NDDS* API documentation (**NDDS.html),** automatically installed in the main directory of your *NDDS* installation.

Reading Guide

If you are new to *NDDS*:

- ❏ Read the sections in Chapter 1 on System Requirements (Section 1.1) and Compatibility with Other RTI Products (Section 1.2).
- ❏ Use the instructions in Chapter 2 to download and install *NDDS*.
- ❏ Start experimenting with *NDDS* by following along with the exercises in the *NDDS Tutorial*. The *NDDS Tutoria*l includes exercises to give you hands-on experience creating, building, and running applications. You can work through these exercises, or just read along and look at the example code supplied with the distribution.
- ❏ Refer to the *NDDS User's Manual* for details on how to use *NDDS* to design, build, and run your applications (in particular, you should read Chapters 1, 2, and 3).
- ❏ Refer to Chapter 3 in this guide when you are ready to compile an application.

If you are upgrading from a previous release of *NDDS*:

- ❏ Read Chapter 1, which includes the following release-specific information: System Requirements, Compatibility with Other RTI Products, and What is New or Changed in NDDS 3.0m since NDDS 3.0i.
- ❏ Use the instructions in Chapter 2 to download and install *NDDS*.
- ❏ Refer to Chapter 3 in this guide for release-specific compiling instructions.

Conventions

Typographical Conventions

This guide uses fonts to clarify text by distinguishing between what you type, what you should see as display output, variables, and literal strings. The typographical conventions are listed in Table 1.

Table 1 **Typographical Conventions**

Usage	Examples
pathnames, files	**/local/rti/ndds**, **Hello_publisher.c**
property names, environment variables	**NDDSHOME**
function names	**NddsPublisherCreate()**
constants, keywords	**NDDS_DEFAULT_DOMAIN**
keyboard input	**`nddsgen -example`**
named keys on keyboard	**<Ctrl>**
display output, source code	**`DBase: started...`**

Pathnames

NDDS documentation often refers to directories and files. For UNIX®-like operating systems, *NDDS* utilities and this guide use forward slashes as path separators, such as **/local/rti/ndds**. Otherwise, we use the backward slash when describing files on the Microsoft® Windows® operating system.

Environment Variables

This guide refers to pathnames that have been customized during installation. Throughout this guide, ***NDDSHOME*** refers to the installation directory of *NDDS*.

Names of Supported Platforms

NDDS runs on several different target platforms. To support the vast array of platforms, *NDDS* separates the executable, library, and object files for each platform into individual directories.

Each platform name has four parts: hardware architecture (such as SPARC® or Motorola® 68020), operating system (such as Solaris™ or VxWorks®), operating system version and compiler. For example, **sparcSol2.8gcc2.95** is the directory that contains files specific to Solaris version 2.8 for the SPARC processor, compiled with gcc version 2.95.

For a full list of supported platforms, see Table 3.1.

Coding Conventions

The following coding convention will help users distinguish, in code, specific concepts:

- ❏ All *NDDS* functions begin with **Ndds**, such as **NddsPublicationCreate()**.
- ❏ All type definitions, structures, and enumerated types begin with **NDDS**, such as **NDDSPublicationProperties**.
- ❏ All C++ classes that are strictly interfaces have the prefix **NDDS** and the suffix **Class**, such as **NDDSDomainClass**. An interface is a class that contains only pure virtual methods. This is a convenient mechanism to describe the user visible methods and hide implementation details.
- ❏ All C++ classes that are derivable and provide implementation have the prefix **NDDS** and the suffix **Derivable**, such as **NDDSDomainDerivable**. These classes can be used by themselves, or you can derive from them to extend their functionality.

NDDS documentation contains many examples. For clarity, parameters that are not relevant will be ignored, i.e. **NddsSubscriptionCreate()**. In some cases, ellipses will be used to show just a few parameters, for example:
NddsSubscriptionCreate(NDDS_IMMEDIATE_SUBSCRIPTION,...).

Note: Ellipses do not represent the C/C++ variable argument syntax.

Chapter 1

Release Notes for NDDS 3.0m

This chapter provides release-specific information and identifies changes and fixes in the latest version of *NDDS*, release 3.0m. If you are new to *NDDS,* you can skip Section 1.3, which is mostly of interest to those who are upgrading.

- ❏ System Requirements (Section 1.1)
- ❏ Compatibility with Other RTI Products (Section 1.2)
- ❏ What is New or Changed in NDDS 3.0m since NDDS 3.0i (Section 1.3)
- ❏ Known Issues in NDDS 3.0m (Section 1.4)

1.1 System Requirements

1.1.1 Supported Operating Systems and Compilers

NDDS requires a multi-threaded operating system. *NDDS* 3.0m supports the target operating systems and compilers listed in Table 1.1.

See Chapter 3 for more information on compilers and linkers.

Note: Additional platforms not listed in Table 1.1 may be supported through special development and maintenance agreements, including LynxOS® 3.1, MIPS®, pSOS®,

Solaris 2.9, VxWorks 5.3.1 and VxWorks AE. Contact your RTI sales representative for details.

Supported Host Platforms:

- ❏ Windows NT® with service pack 5 or higher
- ❏ Windows 2000 with service pack 2 or higher
- ❏ Windows XP
- ❏ Solaris 2.6 and higher
- ❏ Red Hat® Linux® 7.2 and 7.3

Note: Support for Solaris 2.6 and Red Hat Linux 7.2 (2.2 kernel) are deprecated and will be removed in a future release. Also note that not all WaveWorks® tools are available for all operating systems. (*WaveScope®* is not supported on LynxOS.) Contact your RTI sales representative for details.

Table 1.1 **Supported Target Architectures**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Red Hat Linux 7.3 (2.4 kernel)	Intel® Pentium®	gcc 2.96	i86Linux2.4gcc2.96
Red Hat Linux 7.2 (2.2 kernel) (deprecated)	Pentium	egcs 2.91	i86Linux2.2egcs2.91
LynxOS 4.0	Pentium	gcc2.95.3	i86Lynx4.0.0gcc2.95.3
Solaris 2.6 (deprecated)	SPARC	Sun CC 4.2	sparcSol2.6cc4.2
		Sun CC 5.0	sparcSol2.6cc5.0
		gcc 2.7.2	sparcSol2.6gcc2.7.2
Solaris 2.7	SPARC	Sun CC 5.0	sparcSol2.7cc5.0
		gcc 2.7.2	sparcSol2.7gcc2.7.2
		gcc 2.95	sparcSol2.7gcc2.95
Solaris 2.8	UltraSPARC®	Sun CC 5.0	sparcSol2.8cc5.0
		Sun CC 5.2	sparcSol2.8cc5.2
		gcc 2.95	sparcSol2.8gcc2.95

Table 1.1 **Supported Target Architectures**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
VxWorks 5.4/ Tornado® 2.0.x (Solaris or Windows)	ARM7TDMI®	gcc 2.7.9	arm7tdmiVx5.4gcc
	SA-110™	gcc 2.7.9	armsa110Vx5.4gcc
	Intel 486	gcc 2.7.2	i486Vx5.4gcc
	Pentium	gcc 2.7.2	pentiumVx5.4gcc
	PowerPC® 603	gcc 2.7.2	ppc603Vx5.4gcc
	PowerPC 860 (long jump)	gcc 2.7.2	ppc860Vx5.4gcc
	PowerPC 860 (no long jump)	gcc 2.7.2	ppc860Vx5.4gcc.nljmp
	PowerPC EC603	gcc 2.7.2	ppcEC603Vx5.4gcc
	UltraSPARC (SPARCV9™)	gcc 2.96	sparcv9Vx5.4gcc
	Motorola 68020	gcc 2.7.2	m68020Vx5.4gcc
VxWorks 5.4.x/ Tornado 2.1.x (Solaris or Windows)	PowerPC 604	gcc 2.96	ppc604Vx5.4gcc
VxWorks 5.5/ Tornado 2.2 (Solaris or Windows)	Pentium	gcc 2.9	pentiumVx5.5gcc
	Pentium II	gcc 2.9	pentim2Vx5.5gcc
	Pentium III	gcc 2.9	pentium3Vx5.5gcc
	Pentium IV	gcc 2.9	pentium4Vx5.5gcc
	PowerPC 603	gcc 2.96	ppc603Vx5.5gcc
		Diab™ 5.0.1	ppc603Vx5.5diab
	PowerPC 604	gcc 2.96	ppc604Vx5.5gcc
		Diab 5.0.1	ppc604Vx5.5diab
	PowerPC 860	gcc 2.96	ppc860Vx5.5gcc
		Diab 5.0.1	ppc860Vx5.5diab
Windows NT, Windows 2000, Windows XP	Pentium	Visual C++® 6.0	i86Win32VC60

1.1.2 Disk and Memory Usage

Disk usage for a typical installation that supports one host architecture and one target architecture is approximately 38 MB. Installation for more than one target or one host architecture will require more disk space. Typically, each additional target requires 1MB and each host requires 7 MB of disk space.

We recommend that you have at least 64 MB RAM installed on your host development system. Your specific memory requirements, however, will depend on the complexity of your application and hardware architecture.

1.1.3 Networking Support

NDDS requires UDP/IP services. Specifically, *NDDS* needs network socket support for datagrams. These services are part of the standard TCP/IP stack included with most operating systems.

1.2 Compatibility with Other RTI Products

NDDS 3.0m is built on RTILIB 4.2 and therefore is compatible with RTI's *StethoScope*® 7.0, *ScopeTools* 4.0, and *Constellation*™ 1.0 or higher.

NDDS 3.0m is compatible with *NDDS* 3.0i.

NDDS 3.0m is *not* compatible with *NDDS* 2.3 and older versions, due to changes in the messages formats. Running incompatible versions will result in the following problems:

- ❏ Nodes will not be able to communicate with each other.
- ❏ You will not be able to recompile programs.

NDDS 3.0m is *not* compatible with *ControlShell*® (because they use different versions of RTILIB). If you want to use *NDDS* with *ControlShell*, use *NDDS* 3.0g. Contact your RTI FAE or sales representative for more information.

To upgrade existing applications to use *NDDS* 3.0m, compile your application against *NDDS* 3.0m and then use the compiler's error messages to guide you in making the appropriate API changes.

1.3 What is New or Changed in NDDS 3.0m since NDDS 3.0i

This section describes new features and changes in *NDDS* 3.0m compared to *NDDS* 3.0i.

1.3.1 New in NDDS 3.0m

1.3.1.1 New Architectures, Compilers

The complete list of supported architectures appears in Table 1.1. The architectures that are new in this release are:

- ❏ VxWorks 5.5 (and Tornado 2.2)
- ❏ LynxOS 4.0
- ❏ VxWorks 5.4 (and Tornado 2.0) on the Intel 486 processor

1.3.2 Removed or Deprecated in NDDS 3.0m

Wind River System's VxWorks AE operating system, MIPS, LynxOS 3.1, and Linux with kernel 2.2 are no longer supported in the standard product, but may be available by special agreement. Contact your RTI sales representative for details.

See also: Removed as of NDDS 3.0i (Section 1.5), and Removed as of NDDS 3.0g (Section 1.6).

1.3.3 Changes in NDDS 3.0m

1.3.3.1 Multi-NIC Support

- ❏ *NDDS* can support up to 32 NIC interfaces now, instead of 8 in 3.0i. You can use up to 4 of them by setting **NDDS_APP_HOSTS** or **NDDSDomainProperties**. If there are more than 4 valid NIC interfaces, *NDDS* will use the first 4 (not including LOOPBACK) that show up in the NIC interface list.
- ❏ You can use the *nicProperties* field in **NDDSDomainProperties** to disable some specific NIC interfaces.
- ❏ If an **NDDSManager** is spawned by an *NDDS* application, the manager will share the same NIC interfaces with the application.

- *NDDS* is now able to use **NDDS_PEER_HOSTS** file that do not have carriage returns and line feed characters (CR/LF) at the end.
- You can now use **NDDS_APP_HOSTS** to restrict the network traffic within one specific subnet.

  The way you use **NDDS_APP_HOSTS** does not change. You can put a subnet mask, such as 158.78.34.0/ffffff00, in **NDDS_APP_HOSTS** to restrict the network traffic. The difference from 3.0i is that previously the manager used all the available NIC interfaces, so although the application did not send via other subnets, the manager was still using all subnets. In 3.0m, if you start the manager from an application, the manager will restrict its traffic within a specific subnet, just as the application does. The result is that if both applications specify the same subnet, there should be no packets sent via other subnets.

1.4 Known Issues in NDDS 3.0m

1.4.1 Feature Limitations on Specific Operating Systems

- pSOS—Multicast and iteration over publishers and subscribers are not supported. (Note that pSOS is only supported in *NDDS* 3.0g, not in *NDDS* 3.0m.)
- *WaveScope* support is not available for LynxOS targets.

1.4.2 Minimum Separation Problem with Multiple Multicast Subscriptions

There is a known issue with having multiple multicast subscriptions to the same topic with different values of minimum separation.

For instance if you do the following:

1. Create one publication sending every second.
2. Create one multicast subscription with a minSeparation of 2 seconds.
3. Create another multicast subscription (with the same multicast address) with a minSeparation of 3 seconds.

This results in the second subscription receiving data every 4 seconds.

The problem is that the publication keeps track of only the minimum minSeparation value for all subscriptions to the same topic with the same multicast address. In this case this value is 2 seconds, so the publication will only send out data every 2 seconds over the multicast address. The subscription with a minSeparation of 3 seconds will thus reject every other packet because they arrive before the minSeparation period is over.

1.4.3 Unsupported Data Types in nddsgen

The following data types are not supported by **nddsgen**. If you bypass **nddsgen**, you can still use them.

- ❑ unsigned long long
- ❑ long long
- ❑ long double
- ❑ wstring

nddsgen also does not support data types that have an array of structures, and the structure has a class as a member field.

1.4.4 nddsManager Holds onto Sockets Opened by the Spawning Application

When the **nddsManager** is started automatically by the application on UNIX systems (such as when there is no manager running on the appropriate domain when the application is started) it will inherit ownership of the creating application's *NDDS* sockets. These resources will be returned to the system when both the user application and the **nddsManager** terminate.

1.4.5 Polled Subscription Deadline Behavior

Deadline of a polled subscription is checked only when the subscription is polled. Aside from the desired behavior of being notified of deadlines only when you poll, the polled subscription currently exhibits the following behavior:

- ❑ Multiple deadlines are not called; you only get the last deadline.
- ❑ If a new issue is received after a deadline has expired, this deadline is not remembered. When you poll, you simply get the issues that have been queued.

1.5 Removed as of NDDS 3.0i

1.5.1 NDDS_UPDATE_OF_OLD_DATA Event

NDDS silently discards old or duplicate issues without handing them to the user. This affects multi-NIC machines particularly; in *NDDS 3.0g*, applications would get these events if the sending application and the receiving application were in more than one common subnet. While code that uses the event **NDDS_UPDATE_OF_OLD_DATA** will still compile, this event is no longer reported.

1.6 Removed as of NDDS 3.0g

1.6.1 Use of “u_*” types in .x Files Not Recommended

In *NDDS* 2.3, if you specified a type "unsigned xyz" in a **.x** file, **nddsgen** would generate the type "u_xyz" in the **.h** file. However, **nddsgen** no longer behaves this way. Basically, what goes in the **.x** file comes out in the **.h** file. We no longer recommend using types of the format "u_*" in **.x** files.

1.6.2 Deprecated C Creation Functions

The following C creation functions are deprecated and will be removed in the next release. They are still supported in this release.

- **NddsPublicationCreate()**
- **NddsSubscriptionCreate()**
- **NddsClientCreate()**
- **NddsServerCreate()**

Users should switch to these functions:

- **NddsPublicationCreateAtomic()**
- **NddsSubscriptionCreateAtomic()**

- **NddsClientCreateAtomic()**
- **NddsServerCreateAtomic()**

Chapter 2

Installation and Setup

This chapter provides instructions on how to download and install *NDDS*.

- ❏ Overview (Section 2.1)
- ❏ Downloading NDDS (Section 2.2)
- ❏ Installing on UNIX Systems (Section 2.3)
- ❏ Installing on Windows Systems (Section 2.4)

2.1 Overview

NDDS is designed to have all its files under a single tree. For VxWorks users, this chapter assumes VxWorks is already installed and functional. All the directory locations in this chapter are meant as suggestions only; adjust them to suit your site. You do not need administrator privileges to install *NDDS*.

The basic steps described in this chapter include:

1. Downloading a distribution.
2. Unpacking *NDDS* from the distribution.
3. Installing the *NDDS* libraries and utilities.

4. Setting the **NDDSHOME** environment variable (done automatically on Windows).
5. Testing the installation.

Note: If you are using multiple RTI products (such as *Constellation*, *StethoScope*, *MemScope*, and *ProfileScope*) along with *NDDS*, then you should install each product in a separate folder under a common RTI directory; for example:

```
/home/local/rti/ndds.3.0x
/home/local/rti/scope.7.0x
```

2.2 Downloading NDDS

Before you can install *NDDS*, you must download the *NDDS* distribution using the URLs provided in the letter confirming your purchase or evaluation copy. The letter includes your username, password, and the URLs for downloading a distribution that has been built specifically for your architectures. If you do not have this letter, contact **license@rti.com**.

The list below describes the basic steps for downloading. Your letter may contain more up-to-date information tailored for your specific needs. If you need help with the download process, contact **support@rti.com**.

To download NDDS:

1. Download any additional documents and tools:
   a. Go to RTI's home page: www.rti.com.
   b. Select **Downloads**, located at the top of our home page.
   c. Select **NDDS & Tools**, located at the left of the page.
   d. Enter your username and password.
   e. Select **Prebuilt**, located under the **NDDS & Tools** link.

      This page provides links for downloading additional documents and tools. Select the files that you want to download. Any tools that you download from this page should be installed *after* installing *NDDS*.
2. Download *NDDS* using the URLs provided in your confirmation letter.

   Save the files to a temporary directory on your computer, such as **/tmp**.

Distribution Filename Conventions

In the following filename descriptions, ***x*** is a letter identifying the release of *NDDS* 3.0, **<*Arch*>** identifies your host or target architecture family, and **<*ddmmyy*>** is the date the file was built for you.

- **UNIX Systems**—*NDDS* distribution files use the naming convention **Ndds30x<*Arch*>-<*ddmmyy*>.tar.gz**.
- **Windows Systems**—There are two types of *NDDS* distribution files:
  - The basic NDDS executable, **ndds30*x*.exe**.
  - Additional target libraries, **ndds30*x*<*Arch*>.exe**.

2.3 Installing on UNIX Systems

To install a distribution for a UNIX system:

1. Create a directory for *NDDS*. We will assume that the installation directory is **/local**. If you want to install it elsewhere, replace the references to **/local** with the directory of your choice.

   ```
   cd /local
   mkdir rti
   cd rti
   ```

2. Move the downloaded file into your newly created directory. In these instructions, we will assume the name of the file you downloaded is **Ndds30mSolaris28-130803.tar.gz**. (This is only an example filename, yours will be different. The file naming convention is described in Section 2.2.)
3. Uncompress the file. For example:

   ```
   gunzip Ndds30mSolaris28-130803.tar.gz
   ```

4. Extract the distribution by untaring the uncompressed file. For example:

   ```
   tar xvf Ndds30mSolaris28-130803.tar
   ```

5. Add the **bin** directory for your host architecture to your **PATH** environment variable. Assuming you installed *NDDS* into the directory **/local/rti/ndds.3.0x**, type:

   ```
   setenv PATH /local/rti/ndds.3.0x/bin/<architecture>:${PATH}
   ```

   where <architecture> is your specific host architecture, such as sparcSol2.8cc5.0. See Table 1.1 for a list of architectures.

6. Set the **NDDSHOME** environment variable.

   When you untar the distribution, all the files for *NDDS* are located under one directory called **ndds.3.0x** (where ***x*** is a letter identifying the current release of *NDDS* 3.0). This directory is referred to as **NDDSHOME**.

   For each user's "**.login**" file, type:

   ```
   setenv NDDSHOME /local/rti/ndds.3.0x
   ```

   where the directory is the actual location of the *NDDS* files loaded on your system.

   If the **setenv** command is not available, use the **export** command instead:

   ```
   export NDDSHOME=/local/rti/ndds.3.0x
   ```

2.3.1 Testing Your Installation

To test the installation on a UNIX system:

1. Run **rehash** at the command prompt, so the shell will find the new executables.
2. Open two command windows.
3. In one command window, type:

   ```
   nddsPing -s
   ```

   In that window, you should see:

   ```
   nddsPing: deadline
   nddsPing: deadline
   nddsPing: deadline
   ...
   ```

4. In the other command window, type:

   ```
   nddsPing -p
   ```

If *NDDS* is installed correctly, you will see the following in the command window running the **nddsPing -p** command:

```
nddsPing: Sending issue 0
nddsPing: Sending issue 1
nddsPing: Sending issue 2
...
```

You should also see the following in the command window running the **nddsPing -s** command:

```
nddsPing: deadline
nddsPing: received issue 0
nddsPing: received issue 1
nddsPing: received issue 2
...
```

If you do not see issues being sent and received, check the installation procedure and try again.

2.4 Installing on Windows Systems

This section provides instructions on how to install a distribution for Windows NT, Windows XP, and Windows 2000 operating systems.

If you downloaded files for additional target libraries, you will also need to see Section 2.4.1.

To install a distribution for a Windows system:

1. Run the downloaded file, such as **ndds30*x*.EXE**.
2. On the Welcome Window, click the **Next>>** button.

   Note: To exit the installation at any time, click the **Exit Installation>>** button.
3. On the Select Destination Directory Window, select the installation directory where you want to install *NDDS*.

   We recommend that you create a directory where you can keep all of RTI's products. For example, specify **C:\RTI\waveworks\ndds.3.0x** as the installation directory (where ***x*** is a letter identifying the current release of *NDDS* 3.0).

The installation program will set this as your **NDDSHOME** system environment. If you have already defined **NDDSHOME** variable in your own user environment, it will overwrite the one in the system environment variable.

Click the **Next>>** button.

4. On the NDDS Components Window, select all the components you want to install (see Table 2.1).

   Click the **Next>>** button.

Table 2.1 **NDDS Components Window**

Component	Contents
NDDS (required)	Contains the core libraries for *NDDS* and is required to do any development.
Documentation HTML/PDF	Contains online HTML documentation for the API, which can be viewed with any HTML browser (such as Netscape or Internet Explorer) and PDF documentation (this *Getting Started Guide*, the *Tutorial*, and the *User's Manual*) which can be viewed with Adobe® Reader® or Acrobat®.
Examples	Contains source code examples useful to first time *NDDS* users.

5. On the Domain Number Window, specify the default domain number. In most circumstances, use the default 0. This option enables you to start the NDDS Manager from an icon installed on your **Start** menu or Program Manager group.[1]

   Click the **Next>>** button.

6. On the Ready to Install! Window, click the **Start Install** button.

   Note: *NDDS* supplies some DLLs you may already have. In this case, you will receive a warning, click **IGNORE** to go on.

7. Verify that the **C:\RTI\waveworks\ndds.3.0x\bin\i86Win32VC60** directory is in the path by typing **set** at a command prompt.

1. For more information about the *NDDS* Manager, see Section 12.3 in the *NDDS User's Manual*.

2.4.1 Installing Additional Target Libraries on Windows Systems

Before installing any additional target libraries, make sure you have installed the basic *NDDS* distribution for your Windows system (see Section 2.4).

To install additional target libraries:

1. Run the downloaded file, such as **ndds30*x*<*architecture*>.EXE.**
2. On the Welcome Window, click the **Next>>** button.

2.4.2 Testing Your Installation

To test the installation on a Windows system:

1. Open two command prompt windows, so the shell will find the new environment variables.
2. In one command window, type:

   ```
   nddsPing -s
   ```

   In that window, you should see:

   ```
   nddsPing: deadline
   nddsPing: deadline
   nddsPing: deadline
   ...
   ```

3. In the other command window, type:

   ```
   nddsPing -p
   ```

   If *NDDS* is installed correctly, you will see the following in the command window running the **nddsPing -p** command:

   ```
   nddsPing: Sending issue 0
   nddsPing: Sending issue 1
   nddsPing: Sending issue 2
   ...
   ```

   You should also see the following in the command window running the **nddsPing -s** command:

   ```
   ...
   nddsPing: deadline
   nddsPing: received issue 0
   nddsPing: received issue 1
   ```

```
nddsPing: received issue 2
...
```

If you do not see issues being sent and received, check the installation procedure and try again.

2.5 Setup

Besides setting the **NDDSHOME** environment variable (set automatically during installation), you must also set the **NDDS_PEER_HOSTS** environment variable prior to running any *NDDS* application.

NDDS_PEER_HOSTS specifies the scope of *NDDS* communication in a distributed application scenario. It can also restrict communication to only a subset of *NDDS* nodes in the system.

To understand the purpose of **NDDS_PEER_HOSTS**, you first need a basic understanding of what a *manager* is, since this list is used only by managers, not your application. A *manager* is responsible for making sure each application in the system has current information about all the other applications and their objects. It is a program separate from your application that keeps all the applications in your system updated about each other. A manager may either be the *NDDS* manager program that ships with *NDDS* (**nddsManager**), or another vendor's manager (as long as it conforms to the RTPS requirements for managers). By default, the **nddsManager** is started automatically when your application runs.

The hosts in **NDDS_PEER_HOSTS** are merely *potential* fellow managers—they are just hosts that your application's manager should attempt to contact on startup. There is no requirement that there actually be managers on these hosts. By providing a list of fellow managers, you can increase the manager's efficiency during startup. Setting the correct list of peer hosts is mandatory in a system with distributed application management.

For more information on how to run an *NDDS* application, see Chapter 12 in the *NDDS User's Manual*. For more information on managers, see Chapter 11 in the *NDDS User's Manual*.

NDDS_PEER_HOSTS is a colon-separated list of nodes. Each node can be represented by its host name or by its IP address in the standard dot notation. Both unicast and multicast addresses are allowed and any alive fellow managers are assumed to be listening on a well-known port.

For example, to specify that a manager should look for fellow managers on nodes *mars*, *pluto*, and *jupiter,* the peer list could be:

```
mars:pluto:jupiter
```

Or you can use explicit IP addresses as well, for example:

```
206.197.57.101:206.197.57.195:207.110.10.12
```


2.5.1 Setting NDDS_PEER_HOSTS on UNIX Systems

Use the *setenv* command. For example:

```
setenv NDDS_PEER_HOSTS mars:pluto:jupiter
```

Warning: This only sets the environment in the shell and does not affect other shells. To make this the value default for all shells, put the above line in either the **.login**, **.environment**, or the **.cshrc** file or the equivalent file for your system.

2.5.2 Setting NDDS_PEER_HOSTS on Windows Systems

To set the **NDDS_PEER_HOSTS** environment variable in a command prompt window, use the *set* command. For example:

```
set NDDS_PEER_HOSTS=mars:pluto:jupiter
```

Warning: Do not put a space character on either side of the = character.

This sets the environment variable only in the current command prompt window. If you want to set the environment variable for all programs, use one of the following procedures.

Note: Changing an environment variable only takes effects when new programs are started. Existing running programs will not see this change.

On Windows NT Systems:

1. Click **Start**, **Settings**, **Control Panel** to bring up the **Control Panel**.
2. Select the **System** icon to bring up the system properties dialog box.
   a. Select the **Environment** tab.
   b. Type **NDDS_PEER_HOSTS** in the **Variable** field.
   c. Type the host names or IP addresses, separated by colons, (such as **mars:pluto:jupiter**) in the **Value** field.

3. Click **Set** and **Apply**.

On Windows 2000 Systems:

1. Click the **Start** button, **Settings**, **Control Panel** to bring up the **Control Panel**.
2. Double-click the **System** icon to bring up the system properties dialog box, then select the **Advanced** tab.
   a. Click the **Environment Variables...** button.
   b. Click **New...** under the User variable section.
   c. Type **NDDS_PEER_HOSTS** in the **Variable** field.
   d. Type the host names, separated by colons (such as **mars:pluto:jupiter**), in the **Variable Value** field.
   e. Click **OK** and **OK**.

2.5.3 Setting NDDS_PEER_HOSTS on VxWorks Systems

Use the *putenv* command. For example:

```
putenv NDDS_PEER_HOSTS=mars:pluto:jupiter:myTarget
```

Chapter 3

Compiling NDDS Applications

Note: You do not need to read this chapter until you are ready to compile an *NDDS* application.

This chapter provides instructions on compiling *NDDS* applications on the following platforms:

- ❏ Compiling UNIX Applications (Section 3.1)
- ❏ Compiling Windows Applications (Section 3.2)
- ❏ Compiling VxWorks Applications (Section 3.3)

3.1 Compiling UNIX Applications

Before building an *NDDS* application for a UNIX platform, confirm the following setup checklist:

- ❏ *NDDS* 3.0***x*** is installed (where ***x*** stands for the version letter of the current release, see Chapter 1). For installation instructions, refer to Section 2.3.
- ❏ A text editor (such as **emacs** or **vi**) is installed.

- **gmake** is installed. Note: **make** will *not* work. If you do not have **gmake**, you can download it from our web site. From **http://www.rti.com/downloads/download.html**, select Free Downloads from the left-hand navigation bar and select an appropriate download from the next screen.
- The **NDDSHOME** environment variable is set to the root directory of the *NDDS* installation. To confirm, type:

  ```
  env | grep NDDSHOME
  ```

  If not yet set or incorrect, type:

  ```
  setenv NDDSHOME <correct directory>
  ```

- The **NDDS_PEER_HOSTS** environment variable is set to the list of peers[1] that will be communicating with each other. For example, if your machine is called *myComputer* and you want it to talk to two other peers named *venus* and *mars*, type:

  ```
  setenv NDDS_PEER_HOSTS myComputer:venus:mars
  ```

  If you are simply experimenting on a single node (such as during the exercises in the *NDDS Tutorial*), this only needs to be set to *localhost*.

3.1.1 Compiling

To compile an *NDDS* application of any complexity, you can either modify the auto-generated makefile created by running **nddsgen** or write your own makefile. In either case, you must use one of the supported compilers and linkers shown in Table 3.1. Table 3.1 shows the operating system specific compilers, linkers and libraries needed to build *NDDS* applications. For some operating systems we provide several options. For example, on Solaris 2.8 we support three different compilers. You can choose to use either the native 5.0, native 5.2 or gcc 2.95 compilers. The word *native* is used to describe the compiler that you can purchase with the operating system.

You also need to specify *NDDS*-specific compiler defines, include directories and libraries. Also, there is a specific order to the libraries during linking. Table 3.2 and Table 3.3 show the build settings common to all UNIX platforms, for both static and shared libraries (where ***<architecture>*** is the architecture string for your system, as listed in Table 1.1). You must choose between the two types of libraries. Do not mix the different types of libraries during linking.

1. A peer is a computer. It can be denoted as a hostname or an explicit IP address.

The benefit to linking against the shared libraries is that your final executables' sizes will be significantly smaller. You will also use less memory when you are running several *NDDS* applications on the same node. However, shared libraries require more setup and maintenance during upgrades and installations.

Table 3.1 **UNIX Architecture-Specific Build Settings**

Architecture[a]		Compiler (version)	Linker	System Libraries
Solaris 2.6	sparcSol2.6cc4.2	CC (4.2.1)	CC	-ldl -lnsl -lgenIO -lsocket -lgen -lposix4 -lpthread -lm -lc
	sparcSol2.6cc5.0	CC (5.0)	CC	-ldl -lnsl -lgenIO -lsocket -lgen -lposix4 -lpthread -lm -lc
	sparcSol2.6gcc2.7.2	gcc (2.7.2)	gcc	-ldl -lnsl -lsocket -lposix4 -lpthread -lm -lc
Solaris 2.7	sparcSol2.7cc5.0	CC (5.0)	CC	-lnsl -lgenIO -lsocket -lgen -lposix4 -lpthread -lm -lc
	sparcSol2.7gcc2.7.2	gcc (2.7.2)	gcc	-lnsl -lsocket -lposix4 -lpthread -lm -lc
	sparcSol2.7gcc2.95	gcc (2.95.2)	gcc	-lnsl -lsocket -lposix4 -lpthread -lm -lc
Solaris 2.8	sparcSol2.8cc5.0	CC (5.0)	CC	-ldl -lnsl -lgenIO -lsocket -lgen -lposix4 -lpthread -lm -lc
	sparcSol2.8cc5.2	CC (5.2)	CC	-ldl -lnsl -lgenIO -lsocket -lgen -lposix4 -lpthread -lm -lc
	sparcSol2.8gcc2.95	gcc (2.95.2)	gcc	-lstdc++ -ldl -lnsl -lgenIO -lsocket -lgen -lposix4 -lpthread -lm -lc

Table 3.1 **UNIX Architecture-Specific Build Settings**

Architecture[a]		Compiler (version)	Linker	System Libraries
Linux	i86Linux2.4	gcc (2.96)	gcc	-ldl -lnsl -lm -lpthread -lstdc++
	i86Linux2.2	egcs (2.91.66)	egcs	-ldl -lnsl -lm -lpthread -lstdc++-2-libc6.1-1-2.9.0
LynxOS	i86Lynx4.0	gcc (version 2.95.3)	gcc	-ldb -lbsd -lm -lstdc++ -lnsl -lrpc -lc -llynx

a. See Table 1.1 for descriptions of architecture switch strings.

Table 3.2 **UNIX Architecture NDDS Build Settings for Shared Libraries**

Field	Values
Compiler defines	nothing to be defined
Include directories	$(NDDSHOME)/include/unix
Libraries to link with	$(NDDSHOME)/lib/***<architecture>***/libnddsz.so $(NDDSHOME)/lib/***<architecture>***/libnddsutilsz.so $(NDDSHOME)/lib/***<architecture>***/libnddscdrz.so $(NDDSHOME)/lib/***<architecture>***/libutilsxxz.so $(NDDSHOME)/lib/***<architecture>***/libutilsipz.so
Append to LD_LIBRARY_PATH	$(NDDSHOME)/lib/***<architecture>***

Table 3.3 **UNIX Architecture NDDS Build Settings for Static Libraries**

Field	Values
Compiler defines	nothing to be defined
Include directories	$(NDDSHOME)/include/unix
Libraries to link with	$(NDDSHOME)/lib/***<architecture>***/libnddsz.a $(NDDSHOME)/lib/***<architecture>***/libnddsutilsz.a $(NDDSHOME)/lib/***<architecture>***/libnddscdrz.a $(NDDSHOME)/lib/***<architecture>***/libutilsxxz.a $(NDDSHOME)/lib/***<architecture>***/libutilsipz.a

3.2 Compiling Windows Applications

Before building an *NDDS* application for a Windows platform, confirm the following setup checklist:

- ❑ *NDDS* 3.0 is installed. For installation instructions, refer to Chapter 2.
- ❑ Microsoft Visual C++ 6.0 (Service Pack 3) is installed.
- ❑ The **NDDSHOME** environment variable is set to the root directory of the *NDDS* installation. To confirm, type at a command prompt:

  ```
  echo %NDDSHOME%
  ```

  If **NDDSHOME** is set incorrectly, use the instructions in Section 3.2.1 to change its setting.
- ❑ The **PATH** environment variable includes the **bin/i86WinVC60** directory under the **NDDSHOME** directory. The **PATH** can be confirmed and changed using the same instructions given above for the **NDDSHOME** environment variable.
- ❑ The **NDDS_PEER_HOSTS** environment variable is set to the list of peers that will be communicating to each other. If your machine is called *myComputer* and you want it to talk to *venus* and *mars*, type the following at a command prompt:

  ```
  set NDDS_PEER_HOSTS=myComputer:venus:mars
  ```

  If you are simply experimenting on a single node (such as during the exercises in the *NDDS Tutorial*), this only needs to be set to *localhost*.

3.2.1 Setting the NDDSHOME Environment Variable

The **NDDSHOME** environment variable is set automatically to the root directory of the *NDDS* installation, so you will probably not need to change it. If you should need to change it (as may happen if you reinstall *NDDS* in another path), use the following steps.

To confirm the current setting of NDDSHOME, type at a command prompt:

```
echo %NDDSHOME%
```

To temporarily change NDDSHOME in the current command prompt window only, type:

```
set NDDSHOME=<correct directory>
```

Warning: Do *not* put a space on either side of '='. Windows treats this syntax differently.

To permanently change NDDSHOME, use the Control Panel:

On Windows NT:

1. Click **Start**, **Settings, Control Panel**. The **Control Panel** Window pops up.
2. Double-click the **System** icon. The **System Properties** Window pops up.
   a. Click the **Environment** tab.
   b. Under **User Variables**, scroll to the **NDDSHOME** variable and click the line.
   c. In the **Value:** entry field, enter the correct path.
   d. Click **Set**, then click **OK**.

On Windows 2000:

1. Click **Start, Settings, Control Panel**. The **Control Panel** Window pops up.
2. Double-click the **System** icon. The **System Properties** Window pops up.
   a. Click the **Advanced** tab.
   b. Click the **Environment Variables** button.
   c. Under **User Variables**, scroll to the **NDDSHOME** variable and click the line or click the **New...** button.
   d. In the **Variable:** entry field enter **NDDSHOME**. In the **Variable Value:** entry field, enter the correct path.
   e. Click **Set**, then click **OK**.

3.2.2 Compiling

To compile an *NDDS* application of any complexity, you probably need to create your own project file in Microsoft Visual C++ 6.0. *NDDS* requires the following project settings:

If you are using Visual C++ 6.0:

1. Select the multi-threaded DLL project setting:
   a. Select the **C/C++** tab.
   b. From the **Category:** pull-down menu, select **Code Generation**.
   c. From the **Use run-time library:** category, select **Multithreaded DLL**.
2. Link against the *NDDS* libraries:
   a. Select the **Link** tab on the **Project Settings** Window.
   b. From the **Category:** pull-down menu, select the **Input** option.
   c. You have a choice of whether to link against *NDDS*'s static or dynamic libraries. In either case, be sure to use a *space* as a delimiter between libraries, *not* a comma.

      To link against the static libraries: at the *beginning* of the **Object/Library modules:** add the libraries listed in Table 3.4. Or, to link against the dynamic libraries: at the *beginning* of the **Object/Library modules:** add the libraries listed in Table 3.5.
   d. In the **Additional library path:** field, add:

      ```
      $(NDDSHOME)\lib\i86Win32VC60
      ```
3. Specify the path to *NDDS*'s header file:
   a. Select the **C/C++** tab.
   b. From the **Category:** pulldown menu, select the **Preprocessor** option.
   c. In the **Additional include directories: field,** add:

      ```
      $(NDDSHOME)\include\win32
      ```

Table 3.4 **Windows Architecture NDDS Build Settings for Static Libraries**

Field	Values
Compiler defines	nothing to be defined
Include Directories	$(NDDSHOME)\include\win32
Static libraries to link against	$(NDDSHOME)\lib***<architecture>**\libnddsz.lib $(NDDSHOME)\lib***<architecture>**\libnddsutilsz.lib $(NDDSHOME)\lib***<architecture>**\libnddscdrz.lib $(NDDSHOME)\lib***<architecture>**\libutilsxxz.lib $(NDDSHOME)\lib***<architecture>**\libutilsipz.lib
System libraries to link against	ws2_32.lib (supplied by Microsoft Visual C++ 6.0)

Table 3.5 **Windows Architecture NDDS Build Settings for Dynamic Libraries**

Field	Values
Compiler defines	nothing to be defined
Include directories	$(NDDSHOME)\include\win32
Shared libraries with debug symbols to link with	$(NDDSHOME)\lib\i86Win32VC60\libnddsd.lib
Shared libraries without debug symbols	$(NDDSHOME)\lib\i86Win32VC60\libndds.lib
System libraries to link against	ws2_32.lib (supplied by Microsoft Visual C++ 6.0)

3.3 Compiling VxWorks Applications

You can build a VxWorks application by cross-compiling from your development host. First confirm the following setup checklist:

- ❏ *NDDS* 3.0 is installed on the Tornado host. Confirm that the *NDDS* libraries for your VxWorks target platform are in the **lib/** directory[1] under the root directory of *NDDS*.
- ❏ Tornado 2.x development environment is installed.

1. RTI's VxWorks target naming scheme: processor followed by OS and compiler. For example, Power PC 603 running VxWorks 5.4 using the gcc compiler is called ppc603Vx5.4gcc. See Table 1.1.

- ❑ You have a start-up script to set up the target for *NDDS*. For an example script, see **examples/vxWorks/login.cmd** under the *NDDS* root directory.

Required Flag for VxWorks 5.4: You must compile with the flag "**-DRTI_VXWORKS**".

Compiling an *NDDS* application for VxWorks depends on the development platform. For more information, such as specific compiler flags, see the *VxWorks Programmer's Guide*. Tables 3.6 through Table 3.8 show the architecture-dependent flags used to build the *NDDS* libraries. We recommend that you use similar settings.

Table 3.6 **Architecture-Dependent Compiler Settings for VxWorks 5.4/Tornado 2.0.x**

RTI Architecture Abbreviation	Compiler	Compiler Options
arm7tdmiVx5.4gcc	gcc 2.7.9	-mcpu=arm7tdmi -mapcs-32 -mno-thumb-interwork -mno-sched-prolog -fno-builtin -fno-rtti
armsa110Vx5.4gcc	gcc 2.7.9	-mcpu=strongarm110 -mapcs-32 -mno-sched-prolog -fno-builtin -fno-rtti
i486Vx5.4gcc	gcc 2.7.2	-m486 -fno-defer-pop -fno-builtin
pentiumVx5.4gcc	gcc 2.7.2	-mpentium -nostdlib -fno-defer-pop -fno-builtin
m68020Vx5.4gcc	gcc 2.7.2	-m68020-40 -fno-builtin
ppc603Vx5.4gcc	gcc 2.7.2	-mcpu=603 -fno-for-scope -fno-builtin -mlongcall
ppc860Vx5.4gcc	gcc 2.7.2	-mcpu=860 -msoft-float -fno-for-scope -fno-builtin (-mlongcall[a])
ppc860Vx5.4gcc.nljmp	gcc 2.7.2	-mcpu=860 -msoft-float-fno-for-scope -fno-builtin
ppcEC603Vx5.4gcc	gcc 2.7.2	-mstrict-align -nostdinc -fvolatile -fno-for-scope -fno-builtin -mlongcall -msoft-float
sparcv9Vx5.4gcc	gcc 2.96	-fno-builtin -mcpu=ultrasparc -m64

a. optional

Table 3.7 **Architecture-Dependent Compiler Settings for VxWorks 5.4/Tornado 2.1.x**

RTI Architecture Abbreviation	Compiler	Compiler Options
ppc604Vx5.4gcc	gcc 2.96	-mcpu=604 -mstrict-align -fstrength-reduce -fno-builtin -mlongcall

Table 3.8 **Architecture-Dependent Compiler Settings for VxWorks 5.5/Tornado 2.2**

RTI Architecture Abbreviation	Compiler	Compiler Options
pentiumVx5.5gcc	gcc 2.9	-march=pentium -nostdlib -fno-defer-pop -fno-builtin -DTOOL_FAMILY=gnu -DTOOL=gnu
pentim2Vx5.5gcc	gcc 2.9	-march=pentiumpro -nostdlib -fno-defer-pop -fno-builtin -DTOOL_FAMILY=gnu -DTOOL=gnu
pentium3Vx5.5gcc	gcc 2.9	-march=p3 -nostdlib -fno-defer-pop -fno-builtin -DTOOL_FAMILY=gnu -DTOOL=gnu
pentium4Vx5.5gcc	gcc 2.9	-march=p3 -nostdlib -fno-defer-pop -fno-builtin -DTOOL_FAMILY=gnu -DTOOL=gnu
ppc603Vx5.5gcc	gcc 2.96	-mcpu=603 -G 0 -fno-builtin -mlongcall
ppc603Vx5.5diab	Diab 5.0.1	-tPPC603FS:vxworks55 -DTOOL=diab -DTOOL_FAMILY=diab
ppc604Vx5.5gcc	gcc 2.96	-mcpu=604 -G 0 -fno-builtin -mlongcall
ppc604Vx5.5diab	Diab 5.0.1	-tPPC7400FV:vxworks55 -Xkeywords=0x0 -DTOOL=diab -DTOOL_FAMILY=diab
ppc860Vx5.5gcc	gcc 2.96	-mcpu=860 -G 0 -fno-builtin -mlongcall
ppc860Vx5.5diab	Diab 5.0.1	-tPPC860FS:vxworks55 -DTOOL=diab -DTOOL_FAMILY=diab

Cross-compiling for any VxWorks platform is similar to building for a UNIX target (see Section 3.1), except that you need to include **$(NDDSHOME)/include/vx.**

To build a VxWorks application, create a makefile that reflects the compiler and linker for your target with appropriate flags defined. There will be several target-specific compile flags you must set to build correctly. For more information, see the *VxWorks Programmer's Guide*.

Index

B

building
- UNIX applications 3-1
- VxWorks applications 3-8
- Windows applications 3-5

C

compilers supported 1-1
compiling 3-1
- UNIX 3-1
- VxWorks 3-8
- Windows 3-7

D

disk usage 1-4

E

environment variables
- NDDS_PEER_HOSTS 2-8
- NDDSHOME i-xi, 2-2, 2-4, 3-5
- PATH 2-4

I

installing NDDS
- UNIX 2-3
- VxWorks 2-7
- Windows 2-5

L

LynxOS 1-5

M

makefiles 3-10
memory usage 1-4

N

NDDS_PEER_HOSTS 2-8
NddsClientCreate 1-8
NDDSHOME i-xi, 2-2, 2-4, 3-5
NddsPublicationCreate 1-8
NddsServerCreate 1-8
NddsSubscriptionCreate 1-8

P

PATH 2-4
pathnames i-xi

S

supported compilers 1-1

U

UNIX
  compiling 3-1
  installation of NDDS 2-3
unsigned types in .x files 1-8

V

variables i-xi
  NDDS_PEER_HOSTS 2-8
  NDDSHOME 2-2, 2-4, 3-5
  PATH 2-4
VxWorks 1-5
  compiling 3-8
  installation of NDDS 2-7

W

Windows
  compiling 3-7
  installation of NDDS 2-5